Representation and Interactive Editing of Vector Data in Virtual Landscapes

Jürgen DÖLLNER, Konstantin BAUMANN and Henrik BUCHHOLZ

1 Introduction

Vector data offers manifold applications for creating, analyzing, and managing virtual landscapes. Real-time visualization techniques for virtual landscapes have been successfully concentrating on efficient methods to process massive raster data sets yet for a long time, but did not develop specific methods for representing and interactively editing vector data in 3D. In our contribution we present a concept for the representation and interactive editing of 2D vector graphics in real-time 3D virtual landscapes. This concept provides an enabling functionality that allows users to directly interact with objects and processes that are depicted in real-time visualizations of virtual landscapes. In this way, virtual landscapes can become effective direct manipulation interfaces.

Vector data as a fundamental category of geodata is supported by GIS along with its processing and management. It is characterized (Longley et al., 2001) by "the precise nature of its representation method, its storage efficiency, the quality of its cartographic output, and the availability of functional tools for operations like map projection, overlay, and analysis." Using vector graphics we can encode *features* built up by points, lines, and areas.

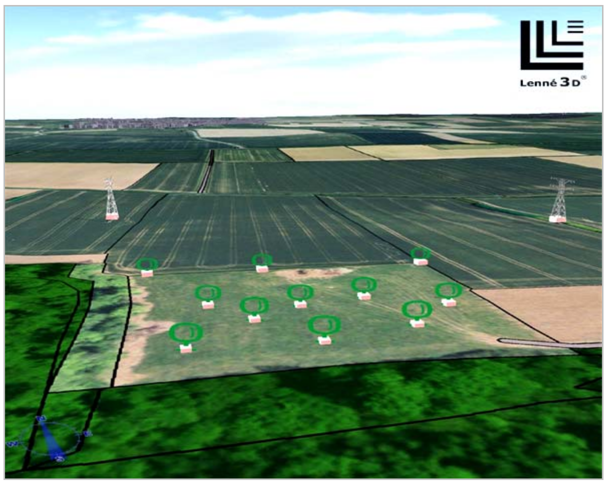


Fig. 1: Real-time virtual landscape visualization depicting land-use areas and positions of planned trees and power poles. (MacEachren, 1995) and 3D map design (Terribilini, 1999).

Typically, vector data is derived from GPS measurements and survey measurements, but it is also a primary data model used to specify information for planning and designing tasks and processes. Vector graphics traditionally plays an important role in 2D map design

For the specification of 3D geovirtual environments such as virtual landscapes, vector graphics models features such as spatial positions, paths, borders, street networks, building ground plans, and land use areas. A typical configuration is illustrated in Fig. 1, which shows a virtual landscape together with an aerial photography and vector graphics overlays depicting land-use areas, tree positions, and power poles.

In classical, two-dimensional map and GIS visualizations, vector data is supported as first-class graphics object. A number of standards emerged such as the Scalable Vector Graphics format (SVG) or industry-driven formats such as Macromedia's flash format (SWF). In 2D, we can directly represent vector data by means of vector graphics primitives; 2D editors provide direct manipulation tools to create and manage vector data. In three-dimensional geovirtual environments, there are a number of difficulties concerning vector data. It has to be adapted to the 3D geometry of the virtual environment (e.g., positioning a polyline over a 3D terrain model) and it has to be integrated with raster data (e.g., superimposing polylines on top of satellite imagery). In the ideal case, vector data objects would adapt themselves to the terrain surface (or any other reference surface), and it should be possible to combine their visualization with the visualization of raster data. In addition, users should be able to directly select and manipulate each individual object in the 3D view.

In our approach, we represent geo-referenced vector data in a hierarchical way based on scene graphs. These scene graphs are rendered in a pre-rendering pass on a per-frame basis in real-time and output 2D textures containing the rasterized vector data. In this way, we can obtain precise, on-demand rasterizations of vector data. The scene graph representation also supports the identification and interactive inquiry of individual vector data objects. Furthermore, the approach allows us to seamlessly integrate vector graphics and raster graphics in 3D geovirtual environments by means of multiple, layered terrain textures.

2 Geometry-Based Depiction of Vector Data

In real-time landscapes, users should be able to directly create and manipulate 2D georeferenced vector graphics such as points, lines, and polygons in order to outline, indicate, or specify spatial locations, paths, or areas. That is, vector data needs to be depicted and manipulated "in situ" in 3D. While geometry-based depictions of vector data do not represent a problem for 2D, these approaches generally cannot be extended to 3D.

To illustrate the problem, consider the following the situation: We want to represent a 2D line segment on top of a virtual 3D terrain surface (Fig. 2a). We could represent the 2D line segment as 3D line segment floating slightly above the terrain surface. For it we would have to tessellate the 3D line segment into a number of sub-segments to adapt it to the terrain surface (Fig 2b). This direct geometric representation, however, could never be

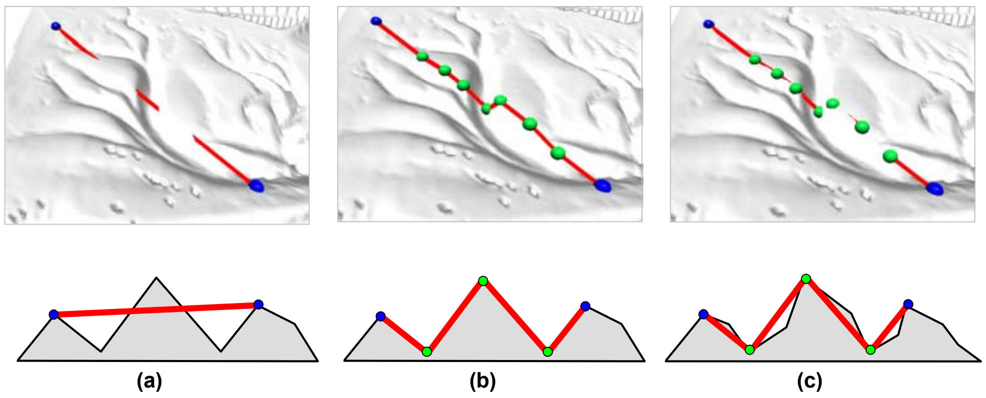


Fig. 2: 3D geometry-based approach for representing vector data and resulting artifacts. (a) Without tessellation. (b) Suitable tessellation. (c) Artifacts after changing the underlying terrain level-of-detail.

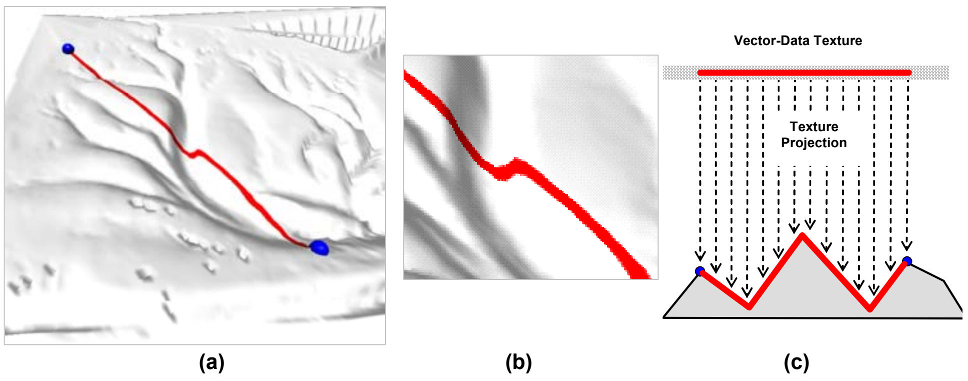
precise because the floating distance depends on the camera distance and screen resolution. In addition, the sub-segments would need to be adapted to the changing level-of-detail terrain surface (Fig. 2c). Even worse, if the sub-segments would be adapted precisely, artifacts would occur due to rendering coplanar primitives.

Similar artifact and quality problems occur in the case of polygons draped over the virtual terrain surface. Furthermore, the 3D geometry-based approach implies additional computational costs due to tessellation and adaptation processes. It also has the disadvantage that vector data can hardly be combined with the raster data visualized by terrain textures. We can conclude that the geometry-based approach of depicting vector data in geovirtual 3D environments has numerous technical drawbacks and severe functional limitations.

3 Texture-Based Depiction of Vector Data

In the texture-based approach of depicting vector data we specify a given collection of vector data objects by a scene graph that represents, hierarchically organizes, and attributes vector data. From the *vector-data scene graph* we derive a texture containing the image of the two-dimensional scene, and we project the texture on top of the reference surface such as the terrain surface. This working principle and an example are illustrated in Fig. 3.

The contents of a vector-data scene graph are rendered into a non-visible framebuffer, using an orthogonal projection during a pre-rendering pass. As the result the framebuffer contains the rasterized representation of the vector data, and the contents are copied into 2D textures using the “render-to-texture” functionality of computer graphics hardware (Akenine-Möller & Haines, 2002, p. 223). These textures can be used like a regular terrain texture in the main rendering pass. The textures can be projected on top of any other raster layers of the terrain such as aerial photography or satellite imagery. Consequently, the



texture-based visualization ensures that vector data can be precisely positioned over the terrain surface.

Fig. 3: Texture-based approach for representing vector data. Terrain-following line segment (a). Zoomed view (b). Working principle using a vector-data texture and projecting it on top of the terrain surface (c).

Fig. 4 shows an example of polygonal data used to mark development areas during a landscape planning process.

The texture-based depiction of vector data can be further enhanced:

- Level-of-detail dependent rasterization of vector data. The vector-data scene graph can be traversed multiple times, each time rendering the vector data in a separate 2D texture to provide rasterizations at different resolutions. For terrain parts close to the camera, smaller areas / higher resolution textures are created. Details about this approach can be found in Kersting and Döllner (2002).

- Annotations. As specialized type of vector data, we can support text in analogy to other 2D shapes. This way, annotations can be integrated into visualizations by textures. The orientation of text can be adapted to the current camera setting to optimize its readability.

Although generally, scene graphs model 3D scenes, we can utilize them to describe two-dimensional scenes. We can also take advantage of 3D for depth-ordering vector data objects, that is, the z-value of 2D vector data objects determines their layering. Scene graphs represent both an effective and efficient technique in computer graphics to encode and render complex geometric data, and they are supported by most real-time graphics libraries as a fundamental functionality of graphics middleware. In this way, we get an efficient and well-known schema for representing and rendering vector data.

4 Direct Manipulation of Vector Data in 3D

Direct manipulation of vector data in 3D allows us to avoid context switches during the interaction of the user with the geovirtual environment, that is, we prevent the user from switching between the 3D view and a 2D editor to create and manipulate vector data. In addition, "in situ" editing allows users to take into consideration geoinformation and thematic information visible in the virtual landscape. For example, users will be more

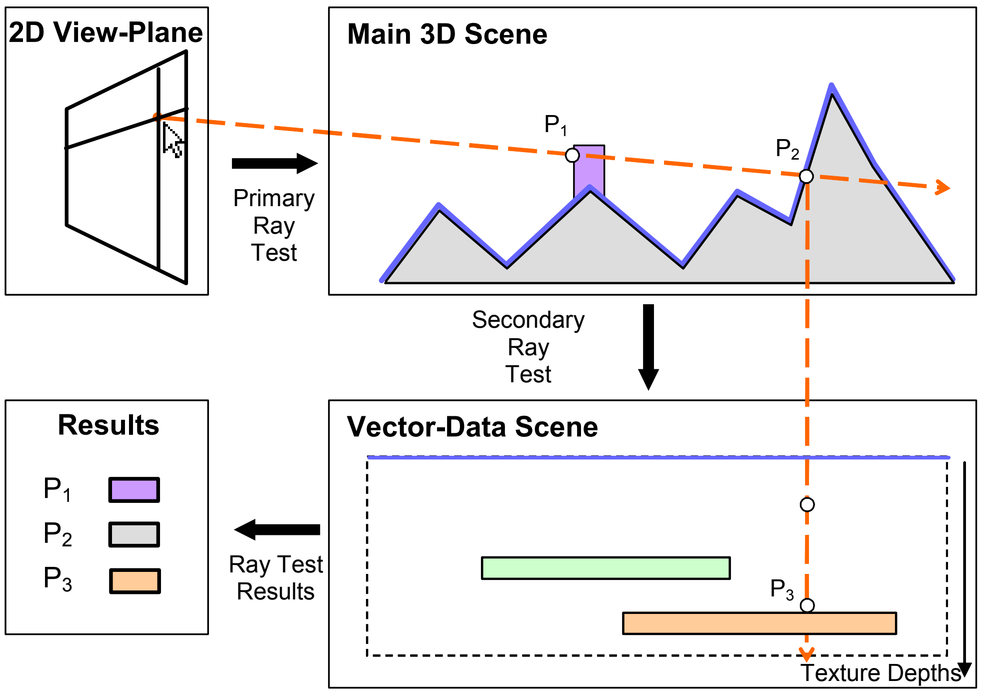


Fig. 5: Picking of vector graphics in 3D scenes. The intersection ray detects intersecting objects in 3D (P_1, P_2) and checks for intersecting 2D vector objects (P_3) if it hits the texture that represents vector graphics using a secondary intersection ray.

effective to outline a building plot in a hilly environment if they can outline the plot on top of the 3D terrain model.

In 3D computer graphics, as Eberly (2001) points out, "the term *picking* refers to the process of selecting a 3D object from its 2D projection on the screen by pointing and clicking with a mouse." For this reasons, picking functionality is a prerequisite for direct manipulation interfaces. A picking operation returns the set of objects that are hit at a view-plane position the user is pointing to. The picking operation can be implemented by testing for intersections between a virtual ray (sent through the view-plane position into the 3D viewing frustum) and the scene objects. In general, this ray test returns a list of depth-sorted hit objects together with the exact 3D intersection points. To implement picking in a hierarchical scene graph, we have to recursively traverse the graph until each leaf node is reached and its geometry tested. In common 3D scene graph systems, ray-based picking is provided as a built-in functionality.

For texture-based vector data, the picking operation starts with a primary ray test: A 3D ray passes the view-plane through the perspective view-frustum (Fig. 5). Along its way, the picking request can hit none, one, or several 3D scene objects. If scene objects are hit, the intersection point, together with the object-id is recorded. If the ray hits the terrain surface, we check for vector-data objects. For it, we send a new, redirected ray through the virtual (2D) scene described by the vector-data scene graph to check for intersections with the visual representations of vector-data objects. The identifiers of hit shapes are returned and, therefore, provide information about individual elements of a vector-data texture.

Using the primary-secondary ray testing schema, we can implement in situ editing operations such as selecting, moving, resizing, rotating, and scaling for vector-data objects similar to 2D editors. In general, it is necessary to add visual handles to the vector-data scene graph (if represented as 2D shapes) or the main scene graph (if represented as 3D objects).

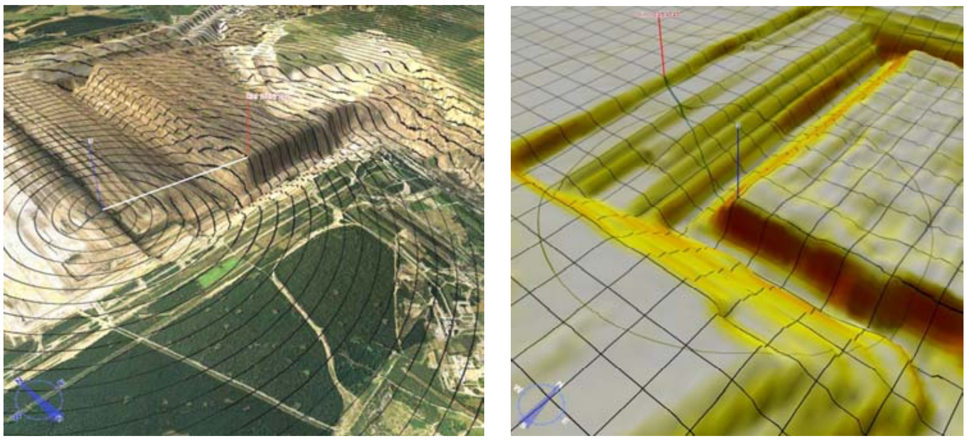


Fig. 6: Using texture-based depiction of vector data to implement interactive visual tools for distance measurements. On the left, concentric circles are used, whereas on the right a regular grid is superimposed to the terrain surface.

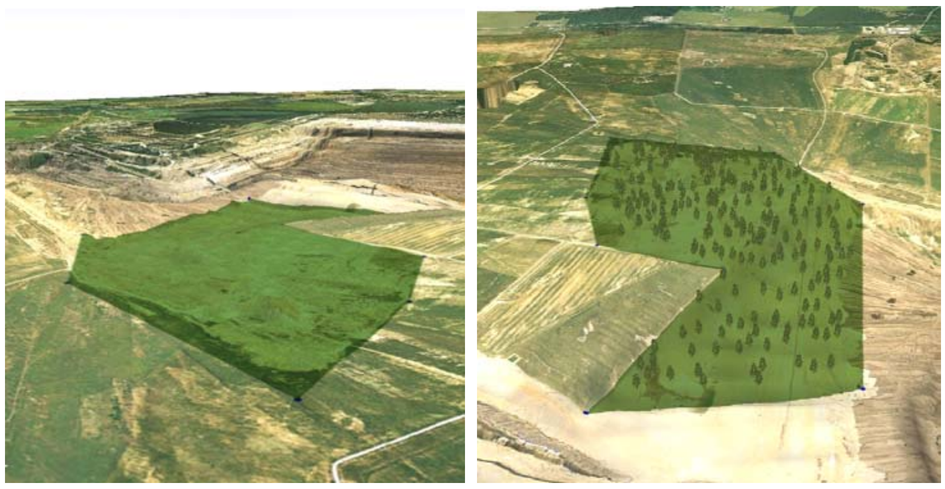


Fig. 7: Using texture-based depiction of vector data to specify plant areas. In the second step, a given number of plants are distributed inside the polygon. Each plant is depicted by a 3D plant model.

5 Analysis & Planning Tools Based on Vector Data

The texture-based depiction of vector data and its direct manipulation provide the functionality to implement analysis and planning tools for virtual landscapes. Vector-data scene graphs can be created instantaneously and modified in real-time, which allows us to blend in temporary, auxiliary vector data into the 3D scene. Two example may illustrate this approach:

- Visual analysis tool for visualizing and measuring distances. In Fig. 6, the user interactively can place a center point and a pattern of concentric circles, respectively a regular grid, implemented as vector-data texture layer, to support visual distance analysis.
- Visual plant distribution tool for creating vegetation objects. In Fig. 7, the user edits a polygon, and then initiates the distribution of a specific number of plants. The plants are depicted by 3D plant models.

6 Conclusions

The texture-based approach for depicting vector data in virtual landscapes in real-time offers an efficient solution for high-quality display and interactive manipulation of vector-data objects in 3D. In particular, it is able to seamlessly integrate vector data and raster data without having to pre-rasterize vector data. The scene graph representation keeps the semantics of vector data accessible in 3D, which is required for any kind of interactive 3D editing and direct manipulation functionality. The implementation also can take full advantage of today's texturing capabilities of graphics hardware.

Vector data in 3D geovirtual environments offers manifold applications such as for the design of analysis and planning tools. Future extensions may address specialized solutions for the texture-based depiction of annotations associated with 3D objects and sophisticated visual tools for supporting the direct manipulation of typical landscape elements.

The presented technique has been implemented as part of the LandXplorer geovisualization system (www.3dgeo.de), which forms part of the Lenne3D project (Paar & Rekittke, 2005). In Lenne3D, vector data turns out to be an essential category of geodata involved in landscape planning processes. In particular, in situ editing of vector graphics is applied to setup vegetation data. The implementation has been based on the Virtual Rendering System VRS (Döllner & Hinrichs, 2002).

Acknowledgements

We would like to thank the German environmental foundation Deutsche Bundesstiftung Umwelt (DBU) for supporting our work within the Lenné3D research project (www.lenne3d.de) as well as our project collaborators.

References

Akenine-Möller, T., E. Haines (2002): *Real-Time Rendering.* 2nd Ed., A K Peters.

Döllner, J., K. Hinrichs (2002): A Generic Rendering System. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):99-118, April-June.

Eberly, D. H. (2001): *3D Game Engine Design.* Academic Press.

Kersting, O., & J. Döllner (2002): Interactive Visualization of 3D Vector Data in GIS. *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems (ACMGIS 2002)*, Washington D.C., 107-112.

Longley, P., M. F. Goodchild, D. J. Maguire, D. W. Rhind (2001): *Geographic Information Systems and Science*. Wiley.

MacEachren, A. M. (1995): *How Maps Work: Representation, Visualization, and Design*. Guilford Press, New York.

Paar, P. & J. Rekittke (2005): Lenné3D - Walk-through Visualization of Planned Landscapes. *Visualization in landscape and environmental planning,* 152-162.

Terribilini, A. (1999): Maps in Transition: Development of Interactive Vector-Based Topographic 3D-Maps. *Proceedings 19th International Cartographic Conference*, 993-1001.